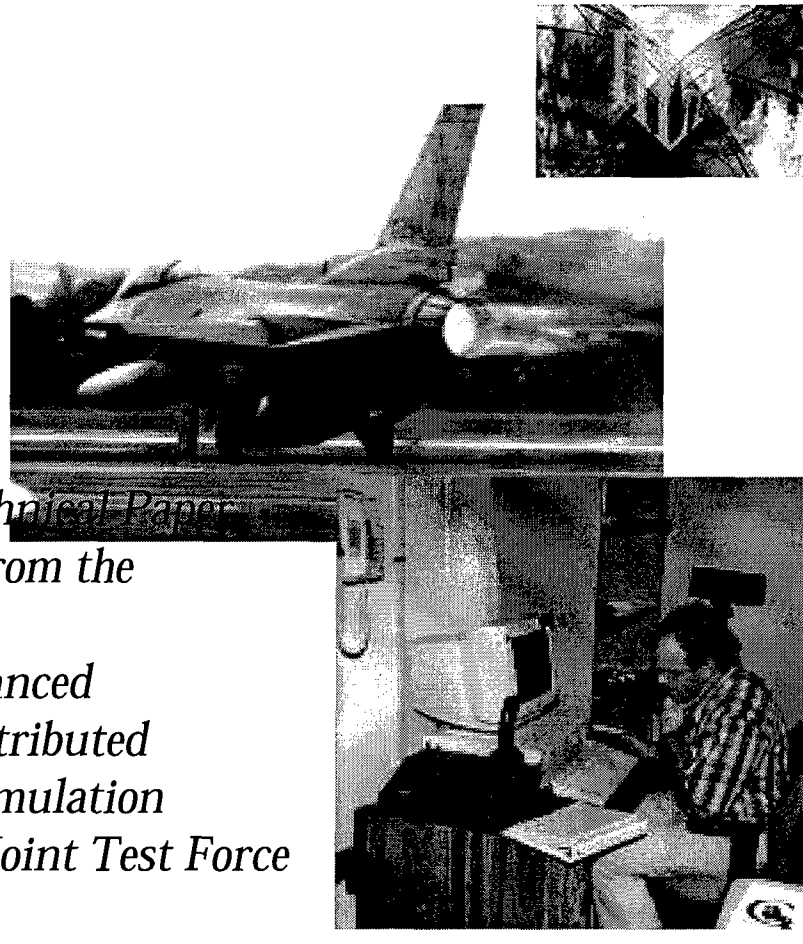
Data Collection in an High Level Architecture Federation



A Technical Paper from the Joint Advanced Distributed Simulation Joint Test Force

Mr. Jerry Black, Science Applications International Corporation

Presented at the International Test and Evaluation Association
Modeling and Simulation Workshop
7-10 December 1998, Las Cruces, NM

JADS JTF
http://www.jads.abq.com
11104 Menaul NE
Albuquerque, NM 87112-2454
(505) 846-1291
FAX (505) 846-0603

Data Collection in an HLA Federation

Mr. Jerry Black
Science Applications International Corporation
JADS JTF
11104 Menaul Blvd NE
Albuquerque, NM 87112
505-846-0467
black@jads.kirtland.af.mil

Keywords:
RTI, HLA, Logger, Data Collection

Abstract: *The Joint Advanced Distributed Simulation (JADS) Joint Test Force (JTF) is chartered by the Office of the Secretary of Defense to investigate the utility of advanced distributed simulation (ADS) technology, including the high level architecture (HLA), for test and evaluation (T&E). JADS is executing three formal test programs, including the Electronic Warfare (EW) Test, representing slices of the overall T&E spectrum to form its conclusions. The EW test is using HLA and Runtime Infrastructure (RTI) version 1.3 to support two electronic warfare test events using a distributed cross-country network linking an instrumented EW system, constructive models, and virtual simulations.*

A method of recording data traffic among federates was required so that accurate latency measurements through the RTI for the JADS federation could be determined. JADS considered three possibilities for a logger – a logger federate, logger capabilities incorporated directly into the federate software, and an interface logger. This paper discusses the differences among the types of loggers, the reasons JADS chose to implement the interface logger, and the implementation of the interface logger.

Background

The JADS EW Test will use high level architecture federations to replicate all elements of an actual open air range test environment and the selected EW system under test (an ALQ-131 Block II self-protection jammer). The JADS federation links six federates on six host computers (Figure 1) passing attributes and interactions within constrained timing tolerances representing EW systems operating in a live test environment. The jammer is represented by the digital system model (DSM) federate residing at the Air Combat Environment Test and Evaluation Facility (ACETEF) facility in Patuxent River, Maryland. The surface-to-air missiles (threats) are represented by the federate at the Air Force Electronic Warfare Evaluation Simulator (AFEWES) facility in Fort Worth, Texas. Four federates – the terminal threat hand-off federate (cues the threat operators); the test control federate; the radio frequency (RF) environment federate (publishes RF background); and the platform federate (publishes aircraft time-space-position information [TSPI]) - reside in the JADS facility in Albuquerque, New Mexico.

An important measurement required to evaluate the ADS environment is the amount of time it takes for the jammer to see the threat and the amount of time it takes for the jammer response to be received by the threat. This total time is referred to as the *federation* latency. To calculate this latency, data must be collected at both the DSM federate and the AFEWES federate.

Overview

The high level architecture (HLA) provides a more flexible environment for linking simulations for distributed tests. Federations are free to define the format of messages exchanged among federates via the federation object model and the simulation object model. However, this flexibility complicates data collection. Due to the fact that there is not a protocol for data exchange (as in distributed interactive simulation protocol data units), a stealth logger can no longer be connected to the network which will recognize and record all simulation traffic in a log file. There are basically three options for data collection in an HLA federation – a logger federate, data collection capabilities incorporated directly into the federate software, and an RTI interface logger.

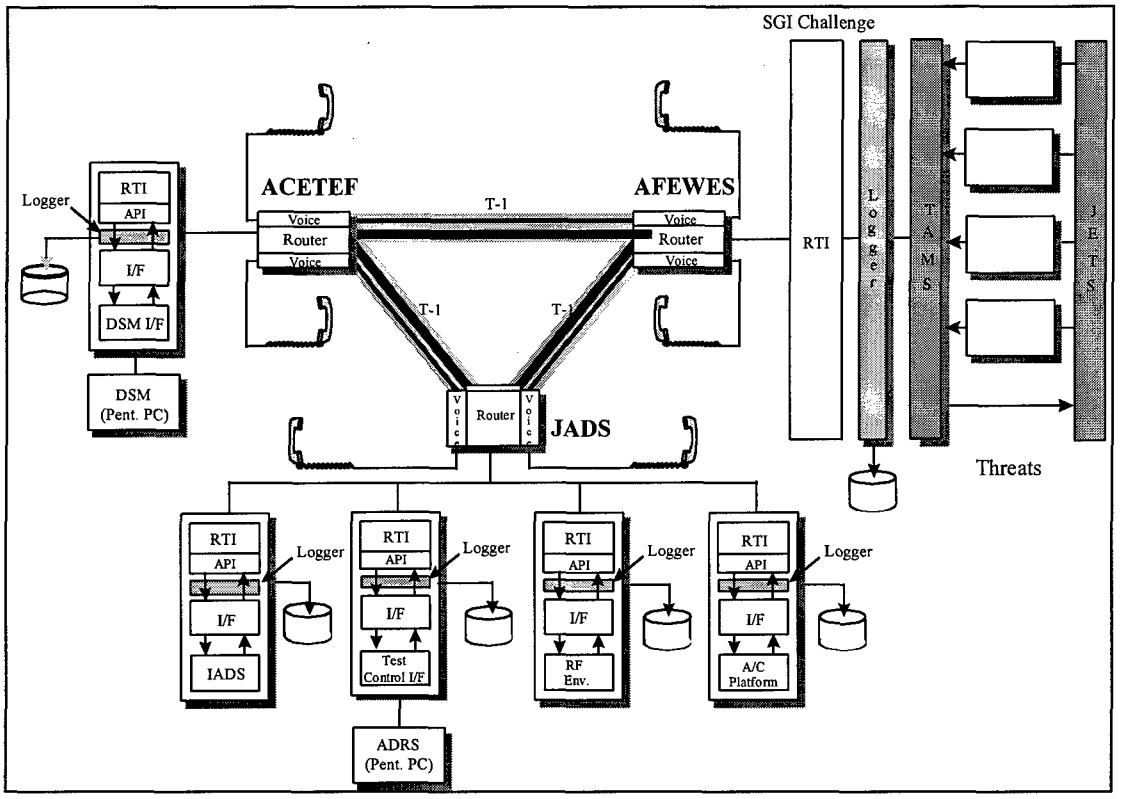


Figure 1. The JADS Federation

A/C = aircraft
API = application program interface
I/F = interface
PC = personal computer
T-1 = digital carrier used to transmit a formatted digital signal at 1.544 megabits per second
TAMS = Tactical Air Mission Simulator
ADRS = Automated Data Reduction Software
IADS = Integrated Air Defense System
JETS = JammEr Techniques Simulator
SGI = Silicon Graphics, Inc.

Logger Federate

A logger federate's sole purpose is data collection. It subscribes to everything and records data as they arrive. If there is only one logger federate in a federation, the network traffic can increase dramatically because the logger federate will have to subscribe to all of the data from the other federates. Another drawback to using a logger federate has to do with the time stamping of the data. The log time associated with the data recorded by a logger federate is the arrival time of the data at that logger federate. There is no indication of the time the data were received by a federate that actually used the data. Therefore, latency calculations become more difficult. To reasonably time stamp the data there would have to be a logger federate at each remote site. The arrival time of data at a given site would be recorded by the local logger federate.

Data Collection Within the Federate

Specialized code can be added to each federate to collect and record data as they are received or published. The main problem with this approach is that code has to be added to every federate in which data are to be recorded. The code will probably be specific to a federate or a federation and cannot be reused by federates in another federation.

RTI Interface Logger

An RTI interface logger resides between the federate software and the application program interface (API) to the RTI and collects all data that pass from the federate to the RTI and all data that pass from the RTI to the federate. JADS chose to implement an RTI interface logger for several reasons. By keeping the logger software separate from the federate

software, it can be optimized to minimize the impact of recording data by the federate. An RTI interface logger can be linked with any federate at any location without additional hardware. Also, since the logger software is not designed specifically for the JADS federation, it becomes valuable as a legacy product after the JADS tests have been completed.

RTI Interface Logger Description

The logger resides in the interface between the federate and the RTI. It records all function calls to and from the RTI along with all of the function data parameters. For example, when the federate wants to publish data, it calls the RTI updateAttributeValues function. When the logger is linked with the federate, the federate calls the logger updateAttributeValues function. The logger stores the function identification and parameter data in the log file buffer and then calls the RTI updateAttributeValues function. When a log file buffer becomes full it is written asynchronously to the log file and a new buffer is created.

Logger Design

The logger was designed to minimize impact on the federate it is linked with. To accomplish this, the logger design includes the following features: asynchronous direct input/output (I/O), nondegrading process priority, and binary file format.

Asynchronous I/O is used so that the federate software does not wait while the data are written to the log file. When a buffer becomes full an I/O request is queued to the operating system and control immediately returns to the federate. The actual writing of the data is accomplished by a separate process.

Direct I/O allows the operating system to use the data buffer created by the logger software to write the data to the disk. Normally, the operating system copies the data from the user's buffer to a system buffer. Use of direct I/O eliminates this copy operation.

If the federate the logger is linked with is executed by a user with super-user privileges, the logger will take advantage of system nondegrading priorities to further minimize the impact of the logger I/O on the federate. When asynchronous I/O is initialized, a set of processes is created to perform the writing of the data to disk. When these processes are created, they inherit the priority of the process that created them. The logger software lowers the priority of the process before the asynchronous I/O is initialized. After the I/O processes have been created, the logger software sets the federate process priority to a real-time priority. Since the I/O processes run at a lower priority than the federate process, the I/O processes will never interfere with the federate process.

The log file created by this software is a binary file. Attribute and interaction data are received by the logger (or by the federate) in a binary format. It takes a series of function calls to translate the attribute handle to a type then convert the binary data to an ASCII format so that the log file could be written in human-readable format. In the interest of minimizing the processing time used by the logger, the binary data received from or sent to the RTI are written directly to the log file without any conversion.

Logger Implementation

The logger software is comprised of three classes. The Logfile_Writer class handles all low-level output to the log file. The two main pieces of software in the logger are the RTI_Logger class and the Federate_Logger class. The RTI_Logger class is derived from the RTI::RTIambassador base class provided with the RTI. The RTIambassador is how a federate communicates with the RTI. Every method in the RTIambassador class has a corresponding method defined in the RTI_Logger class. In the same way, the Federate_Logger class is derived from the RTI::FederateAmbassador base class. The FederateAmbassador is used by the RTI to communicate with the federate.

Since the logger software writes all of the binary data sent to or received from the RTI without attempting to translate or convert them, the logger can be linked with any federation without any modifications to the logger software. Also, since the logger classes are derived from the RTI base classes, very few lines of code must be changed to incorporate the logger into an existing federate. Less than twenty lines of code were modified or added to link the logger with the helloWorld demo program provided with the RTI.

Limitations

Since the logger is derived from classes provided by the RTI, an implementation of the logger is specific to an implementation of the RTI. JADS has developed versions of the logger that work with RTI 1.0v2 and RTI 1.3v4.

Although, the data can be recorded by the logger software without modification, federation-specific reader software must be developed to read the data from the log files. JADS has developed software to read the JADS federation attributes and interactions from the log files as well as general purpose software that reads nonfederation specific data from the log files (e.g., RTI method names and RTI parameters).

The logger software was developed for the JADS federation. The JADS federates execute on Silicon Graphics, Inc., (SGI) computers. To minimize the impact of the logger on the federate, the logger software uses some SGI-specific functions to change process priority and use asynchronous direct I/O.

Log File Reader Software

Since the file created by the logger is a binary file, software is required to read the binary data and convert them into a readable format. JADS has developed programs to extract data of interest to the JADS analysts from the log files and create reports from the extracted data.

Logfile_Summary

The logfile_summary program creates a report containing summary statistics for a log file. The report contains the name of the federate and the start time from the log file. Then for each attribute and interaction (data type) sent or received by the federate, the report lists (by source) the number of updates, the minimum, maximum, and mean latency for the data type.

Display_Time

For every attribute and interaction sent or received by the federate, the display_time program displays all of the time information associated with the update. The raw log time (seconds since 1970 and microseconds), the log time converted to milliseconds since midnight, the header time (in milliseconds since midnight), and the latency for each update are displayed.

Mode_Changes

The mode_changes program is used to show all of the threat mode changes and jammer responses that occurred during the test. The mode and code value are displayed along with the time generated and the time logged. The *federation* latency is the time from when a mode is generated by a threat until the time it is received by the jammer plus the time from when a jammer response is generated by the DSM to the time it is received by the threat. The output from the mode_changes program will be used to determine these values.

Create Analysis File

The Analysis and Data Reduction System (ADRS), created by Georgia Tech Research Institute, is an important analysis tool used by JADS. Its primary function is to display the threat and TSPI data real time during a test. However, it is also an important post-test tool. For post-test analysis, it accepts comma-delimited files containing the attribute and interaction data. These data files are normally generated from the raw range data using a script generator. JADS has developed the create_adrs_file program that extracts attribute and interaction data from a log file and creates a comma-delimited text file that can be read into ADRS.

File Comparison

An important part of JADS EW Test analysis is verification of the accuracy of the federate software. The federates publish their data from scripts. The scripts contain comma-delimited values for the attributes and interactions that are to be published. Analysis files generated by the create_adrs_file program are in the same format as the scripts. The file_compare program compares selected data from two files (either a script file and a log file; or two log files) and determines whether the selected data match. This program will be used to determine if the data a federate published match the data in its script. It will also be used to determine if the data a federate receives are the same as the data another federate published.

RTI Interface Logger Latency Tests

These tests determine the latency added by the JADS RTI interface logger (the logger) used in the JADS EW Test federation. Remember that the logger resides in the interface between the federate and the RTI. All communication to and from the RTI is

recorded in the log file. In most cases logging of calls to *tick* is not necessary. So for this test (and for the actual JADS EW tests) the logger will be run with the NO_TICK option. This means that all function calls (or methods) except *tick* will be logged.

The logger uses asynchronous direct I/O to write buffers to the log file. This is the most efficient method to write a file with minimal impact on the main process. To further minimize the impact of

Maximum Latency	platform		tcf	
	attributes	interactions	attributes	interactions
Without the logger	15 - 23 ms	15 - 26 ms	21 - 30 ms	15 - 25 ms
With the logger (user)	18 - 45 ms	17 - 22 ms	23 - 43 ms	14 - 26 ms
With the logger (root)	16 - 22 ms	15 - 22 ms	17 - 29 ms	14 - 25 ms

Figure 2. Logger Latency Test Results

writes to the log file, the logger sets the priority of the I/O processes to a low value. Then it sets the priority of the main process to a real-time value. To take advantage of this feature, the federate linked with the logger must be executed from a user account that has super-user privilege (e.g., root).

The *testfed* federate was used for the logger latency tests. This federate accepts command line arguments that specify the characteristics of an instance of the federate. The user can specify the federate identification number (-f), the duration of the test (-d), the size of the attributes and interactions (-s), the rate that attributes are published (-r), the number of updates at the specified rate (-n), the amount of time the federate should wait before starting to publish at its specified rate (-w), and whether interactions should be published (-i). There is an additional argument (-c) that indicates which federate is the controller. There must be one and only one controller federate in the *testfed* federation. There are only one attribute and one interaction used by all federates. All federates subscribe to the attribute and the interaction. The attribute is published best effort. The interaction is published reliable.

The tests were performed on the tcf and platform SGI O2s in the Test Control and Analysis Center. The test attribute and interaction sizes and publish rates were selected to closely match the sizes and rates that will be published by the JADS federation. The federate executing on the platform O2 published 11 attribute updates at 20 hertz (Hz) with a size of 121 bytes. The federate executing on the tcf O2 published 2 attribute updates at 20 Hz with a size of 121 bytes. Both federates published interactions at 2 Hz. Each test ran for 60 seconds. The federation executive executed on tcf.

There were three test cases executed. In the first, the federates executed without the logger. In the second, the logger was used but the federates were executed as a normal (not root) user. In the third, the logger was used and the federates were executed as a root user. Each test case was run ten times. (See Figure 2.)

Running with the logger as a non-super-user can, in some cases, add latency. But when a federate linked with the logger is executed using super-user privilege, the latency is not noticeable. There were no data losses in any of the runs in any test case.

Conclusions

The RTI interface logger developed by JADS provides a simple method of recording data by any federate in an HLA federation with some relative advantage over the alternatives. Data can be collected at any location where a federate resides. The logger software does not need to be modified to execute within other federations besides JADS. Linking the logger software with a federate requires few modifications to the federate software and has a minimal impact on the performance of the federate.

Author Biography

Jerry Black is a senior software engineer for SAIC in Albuquerque, New Mexico. He received a B.S. in Computer Science from the University of New Mexico in 1981. Mr. Black has more than 17 years experience developing software for Department of Defense applications including flight software testing, imagery intelligence training, B-2 follow-on T&E, and real-time data collection. He has been supporting the JADS Joint Test Force as a software analyst responsible for time synchronization, data collection and analysis since 1995.

INTERNET DOCUMENT INFORMATION FORM

A . Report Title: Data Collection in an High Level Architecture Federation

B. DATE Report Downloaded From the Internet 1/28/99

C. Report's Point of Contact: (Name, Organization, Address, Office Symbol, & Ph #): **Joint Advanced Distributed Simulation**
Joint Test Force
ATTN: Ann Krause (505) 846-1291
11104 Menaul NE
Albuquerque, NM 87112-2454

D. Currently Applicable Classification Level: Unclassified

E. Distribution Statement A: Approved for Public Release

F. The foregoing information was compiled and provided by:
DTIC-OCA, Initials: VM_ **Preparation Date**: 1/28/99__

The foregoing information should exactly correspond to the Title, Report Number, and the Date on the accompanying report document. If there are mismatches, or other questions, contact the above OCA Representative for resolution.